A Predictive Approach for the Efficient Distribution of Agent-Based Systems on a Hybrid-Cloud

Chahrazed Labba, Narjes Bellamine Ben Saoud, Julie Dugdale

A Predictive Approach for the Efficient Distribution of Agent-Based Systems on a Hybrid-Cloud

Chahrazed Labba[a,*], Narjès Bellamine Ben Saoud[a], Julie Dugdale[b,c]

[a] *Laboratoire RIADI-Ecole Nationale des Sciences de l'Informatique, Univ Manouba, Tunisia*
[b] *Laboratoire d'Informatique de Grenoble*
[c] *University Grenoble-Alps, France, University of Agder, Norway*

Abstract

Hybrid clouds are increasingly used to outsource non-critical applications to public clouds. However, the main challenge within such environments, is to ensure a cost-efficient distribution of the systems between the resources that are on/off premises. For Multi Agent Systems (MAS), this challenge is deepened due to irregular workload progress and intensive communication between the agents, which may result in high computing and data transfer costs. Thus, in this paper we propose a generic framework for adaptive cost-efficient deployment of MAS with a special focus on hybrid clouds. The framework is based mainly on the use of a performance evaluation process that consists of simulating various partitioning options to estimate and optimize the overall deployment costs. Further, to cope with the irregular workload changes within a MAS and dynamically adapt its initial deployment, we propose an extended version of the Fiduccia-Mattheyses algorithm (E-FM). The experimental results highlight the efficiency of E-FM and show that an efficient MAS deployment to hybrid clouds depends on various factors such as the cloud providers and their different cost-models, the network state, the used partitioning algorithm, and the initial deployment.

Keywords: Agent-Based System, Graph partitioning algorithm, Deployment, Hybrid Cloud, Prediction Process, Metrics.

*Corresponding author.
Email address: chahrazedlabba@gmail.com (Chahrazed Labba), narjes.bellamine@ensi.rnu.tn (Narjes Bellamine),Julie.Dugdale@imag.fr (Julie Dugdale)

1. Introduction

Multi-agent systems (MAS) are widely used to model and simulate scalable software systems in many research domains [1][2][3]. Due to their distinct capabilities such as autonomy, adaptability and flexibility, agents represent an attractive solution for building distributed applications. However, the main problem with such applications is the irregular workload as well as the intensive communication between the agents, which may inhibit the efficiency of the underlying infrastructure used to run the system. Further, to cope with the significant workload variations, to reduce delays (long simulation times), and to meet new system requirements in terms of computing and memory resources, a powerful computing infrastructure is required for a proper execution. Unlimited computational resources are not in the reach of all organizations and budget constraints always remain a strong consideration. Thereby, a flexible on-premise infrastructure that can expand and shrink based on the computational requirements of the application is required. As hybrid cloud computing has become a popular architecture where systems are built to take advantage of both public and private infrastructures to meet different requirements [4], it presents a good target environment for the aforementioned challenge.

A hybrid cloud is a cloud computing environment that integrates both on-premise and public cloud resources. The use of such a cloud model can be tackled from two perspectives: (i) a privacy perspective, in which an organization can process its critical data in private and process the non sensitive operations in public, and (ii) a bursting perspective, where the local resources are insufficient, thus extra public cloud resources are required to handle the workload peaks. In our work, we use the hybrid cloud to overcome the challenge of scalability that can be induced by using only the private infrastructure and to minimize the costs of outsourcing the whole data and computing on public resources.

To make an efficient use of such environment, the agents in the MAS must be efficiently deployed across the hybrid cloud resources so that the overall deployment costs are reduced. Since the MAS requirements in terms of computational resources may be subject to continuous changes during run-time due to workload changes, the deployment may need to be continuously adapted by: (i) allocating additional public cloud resources and/or de-allocating existing ones as well as (ii) ensuring a load-balancing over the used resources. However, given the diversity of both cloud providers and their offered resources, the continuous changes in the entire system workload, the frequent load-balancing issues and

the intensive communications within the MAS, the deployment may result in high computation and data transfer costs. Consequently, achieving an adaptive cost-efficient MAS distribution across hybrid clouds is a challenging task. Efficiency is expressed in terms of reduced deployment costs while fulfilling the system requirements in terms of memory and computing resources.

In the frame of this work we propose:

- A generic framework for adaptive cost-efficient deployment of MAS to a hybrid cloud. The framework is based mainly on the use of a performance evaluation process that takes as input: (i) a MAS with varying demands for computational resources and (ii) a hybrid cloud with its diverse on/off premise resources. The evaluation process then estimates and optimizes the overall deployment costs by allocating the right public cloud resources and minimizing the communication costs between the private and public clouds. The estimation and optimization are performed based on the changes in the MAS requirements in terms of memory and computing resources due to the dynamic changes in the agents' workload during their execution.

- An extension of the Fiduccia-Mattheyses (E-FM) graph based partitioning algorithm to cope with the problem of finding a hybrid cloud deployment with reduced costs. As communication is a key challenge in cloud environments, E-FM focuses mainly on minimizing the cost of the exchanged data between the private and public cloud resources.

The implemented approach highlights the efficiency of the proposed E-FM algorithm and shows that the final deployment costs of an agent-based system with varying demands in terms of computational resources is sensitive to multiple factors such as the cloud providers and their different cost-models, the network state, the used partitioning algorithm, as well as the executed scenarios.

Thus far, the distribution of MAS towards more scalable systems has been widely addressed through the use of clusters [5] [6] and grids [7] [8]. Whereas, few approaches have been elaborated to deploy MAS on resources within a single cloud provider [9] [10]. To the best of our knowledge, this is the first approach that focuses on deploying MAS with varying demands for computational resources on a cloud environment that can incorporate multi-public cloud providers.

The rest of the paper is organized as follows: section 2 presents a motivating example. Section 3 gives an overview of the basic concepts. Section 4

describes the proposed generic framework for deploying MAS to hybrid clouds. Afterwards, section 5 highlights the experimental results. Related work and conclusions are presented in sections 6 and 7 respectively.

2. Motivating example

MAS have been widely used to model, simulate and test various evacuation and coordination strategies within disaster management domain [11] [12] [1]. Using crisis evacuation as a scenario, we consider the MAS shown in Figure 1, which consists of $\mathcal{N}$ agents moving around in the spatial environment of a city.

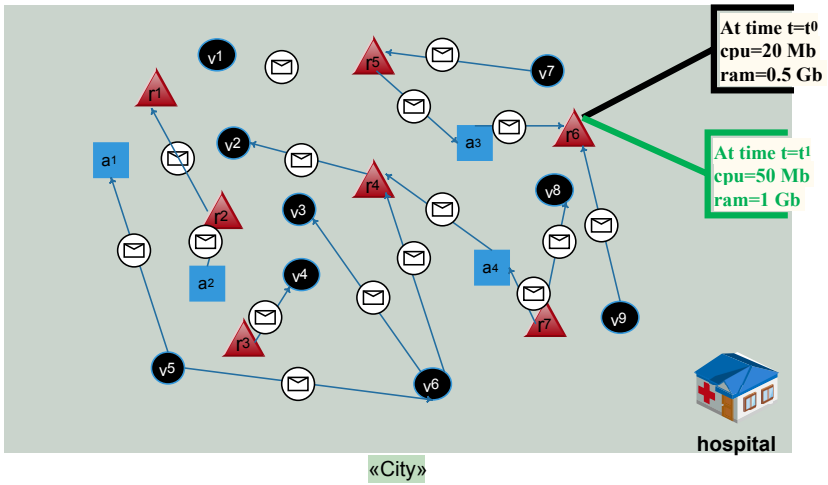


Figure 1: Agent-based Model

The model contains three types of agents including victims shown as black ellipsis, rescuers presented as red triangles, and ambulances/transport vehicles shown as blue squares. The victim agent has three possible states: (i) *normal* injury that can be treated on the accident site, (ii) *serious* injury that can be treated in the hospital and (iii) dead if the agent is seriously injured and no rescuer appears. The rescuer agent retrieves a victim after receiving a notification about its location. The ambulance agents are responsible for carrying the seriously injured victims to the hospitals.

Each agent has a set of requirements in terms of memory and computational resources at start time $t{=}t_0$. Those requirements change during run-time based on the workload executed by the agent.

For example, initially at t=t_0, the agent r_6 needs to be assigned to a Virtual Machine (VM) with a minimal memory (0.5 Gb) and a minimal CPU (20 Mb). Whereas at $t = t_1$, r_6 requirements in terms of memory and computational resources change to a minimal memory of (1 Gb) and a minimal CPU of (50 Mb).

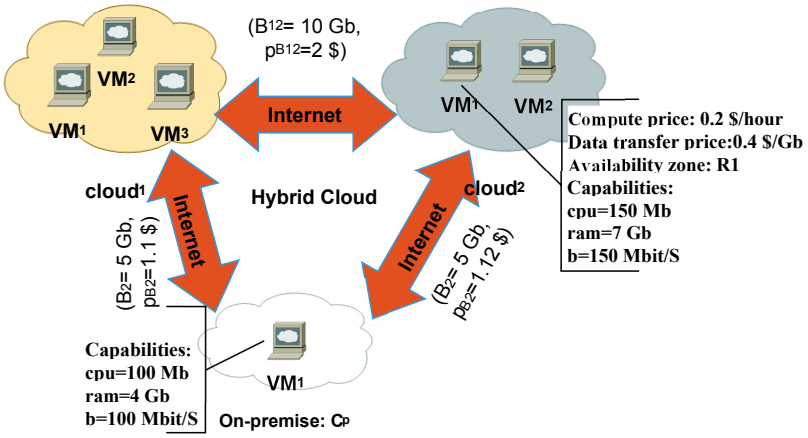
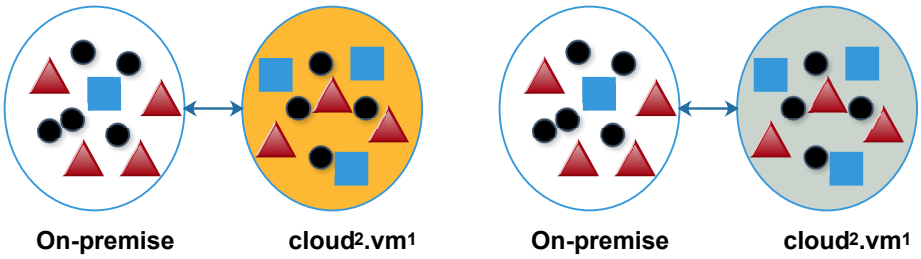


Figure 2: Hybrid Cloud

The hybrid cloud shown in Figure 2 integrates the on-premise resources within a research organization and two public cloud providers $cloud_1$ and $cloud_2$. Each public cloud provider hosts a set of VMs. For example $cloud_1$ has three VMs= (vm_1, vm_2, vm_3). Each VM is characterized by its compute and data transfer prices, its availability zone as well as a set of capabilities including CPU, RAM and bandwidth. For example vm_1 within $cloud_2$ has a compute price of (0.2 \$/hour), a transfer data price of (0.4 \$/Gb) and an availability zone in the region R1. Moreover, it has a CPU of (150 Mb), a (7 Gb) of RAM and a bandwidth of (150 Mbit/s).

A suitable MAS deployment to the hybrid cloud consists of (i) starting with an appropriate initial distribution of the MAS that minimizes the deployment costs, and (ii) adapting the resource allocation to the changes in the demands for memory and computational resources of all agents within a MAS. Assume that the current on-premise infrastructure is insufficient in terms of available memory and computational resources to run the MAS shown in Figure 1. Let S_1 and S_2 illustrated respectively in Figure 3a and Figure 3b be two possible initial deployment solutions for the MAS across the hybrid cloud of Figure 2.

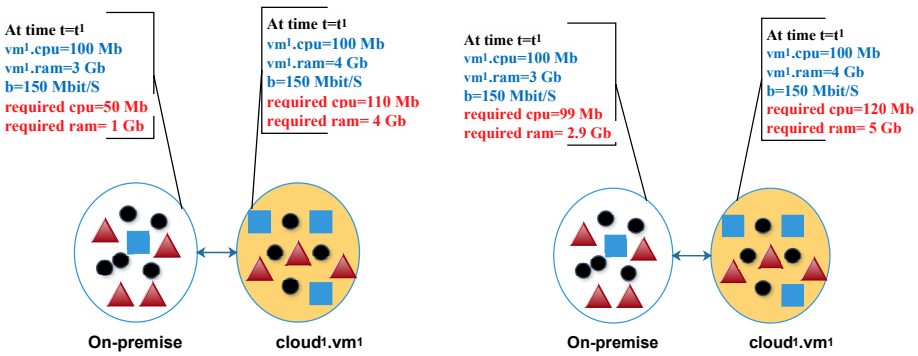
Assuming S_1 costs (6\$) for compute and data transfer costs, whereas S_2 is



(a) Initial deployment solution S_1. (b) Initial deployment solution S_2.

Figure 3: Initial Deployment of the MAS to the hybrid cloud.

charged for (8$) for overall deployments costs. Thereby, among all the possible initial deployment solutions, we are interested in achieving a cost-efficient deployment that reduces the compute and data transfer costs. Assume that S_1 presents the suitable initial deployment at $t{=}t_0$. At $t{=}t_1$, with the dynamic changes in terms of required CPU and RAM, one of the two following cases may arise:



(a) Imbalanced partitions. (b) Insufficient public cloud resources.

Figure 4: Dynamic changes of the agents requirements at $t = t_1$.

- **Imbalanced partitions**: as shown in Figure 4a, the virtual machine within the provider $cloud_1$ is overloaded, in other words the required CPU exceeds the capabilities of the VM, while the on-premise VM is underloaded. In this case, a redistribution of the agents within the MAS is sufficient to solve the problem. However, while migrating a set of agents from the overloaded partition to the opposite side, the communication costs need to be kept minimized.

- **Insufficient used resources**: as shown in Figure 4b, the current leased cloud resource is insufficient in terms of CPU and RAM to execute the hosted agents. Due to the minimal available CPU and RAM on-premise resource, a redistribution of the agents can not solve the problem. Thus new public cloud resource allocations are required to properly execute the MAS.

We are interested in providing a cost-efficient adaptive deployment solution over time of a given MAS to a given hybrid cloud. Therefore an offline approach consisting of predicting and optimizing the overall deployment costs is required.

The next section provides an overview of the basic used concepts. Afterwards the proposed approach is presented.

3. Preliminaries and Background

In this section, we provide an overview of the basic used concepts in this work including (i) hybrid cloud, (ii) MAS and, (iii) adaptive deployment of MAS on hybrid clouds.

3.1. Hybrid Cloud

A hybrid cloud [13] refers to the integration of both *private* and *public* cloud models via defined bandwidths. In our work, we use a hybrid infrastructure to expand the on-premise resources with public ones to ensure a proper execution of the MAS over time. We define the on-premise infrastructure , the public cloud provider and the hybrid cloud respectively in definitions 1,2 and 3.

Definition 1: (On-premise Infrastructure) An on-premise infrastructure $\mathscr{I}_p$ is defined as a tuple (V_M) where:

$V_M = \{vm_i \mid i \geq 1\}$ represents the set of virtual machines available within $\mathscr{I}_p$. Each vm is defined as a tuple $vm = (mr, mc, mb)$ where the terms correspond respectively to maximum memory capacity (Gb), maximum compute capacity (Gb), and the maximum bandwidth capacity (Mb/s).

Definition 2: (Public cloud provider) A cloud provider $\mathscr{C}_p$ is defined as a tuple (VM) where:

$V_M = \{vm_j \mid j \geq 1\}$ represents the set of virtual machines provided within $\mathscr{C}_p$. Each vm is defined as a tuple $vm = (pc, pd, a, mr, mc, mb)$ where the terms correspond respectively to the compute price ($/hour), data transfer price ($/GB), availability zone, maximum memory capacity (Gb), maximum compute capacity (Gb), and the maximum bandwidth capacity (Mb/s).

Definition 3: (Hybrid cloud) A hybrid cloud $\mathscr{H}_{\mathscr{C}}$ is defined as a tuple (Cap_{H_C},$\mathscr{C}$) where:

- $Cap_{H_C} = (\{B_i, pb_i\} \mid i \geq 0)$ represents the characteristics of $\mathscr{H}_{\mathscr{C}}$ in terms of used bandwidths (Gb/s) and its prices ($/hour).
- $\mathscr{C}$ represents both on-premise resources $\mathscr{I}_p$ and the set of cloud providers $\mathscr{C}_p$ within $\mathscr{H}_{\mathscr{C}}$.

An example of a hybrid cloud $\mathscr{H}_{\mathscr{C}}$=($Cap_{H_C}$,$\mathscr{C}$) is depicted in Figure 2 where, Cap_{H_C}= ($\{B_1, pb_1\}$, $\{B_2, pb_2\}$, $\{B_{12}, pb_{12}\}$) such that $\{B_1, pb_1\}$= (5 Gb, 1.1 $), $\{B_2, pb_2\}$=(5 Gb, 1.12 $) and $\{B_{12}, pb_{12}\}$=(10 Gb, 2 $). Further, $\mathscr{C}$=($cloud_1$, $cloud_2$, C_p); $cloud_2$=(V_M); V_M = (vm_1, vm_2); vm_1=(0,2 $/hour,0,4 $/Gb, R_1, 150 Mb, 7Gb, 150 Mbit/S); C_p=(VM); VM=(vm_1); vm_1=(100Mb, 4 Gb, 100 Mb/s)

3.2. Multi-Agent System

A multi-agent system consists of a set of *agents*, which *interact* with one another via *messages*, in a common *environment*(real/virtual) where they can act and cooperate to achieve system objectives [14] [15] [16]. We define an agent and a MAS respectively in Definition 4 and 5.

Definition 4: (Agent) Let $ag \in \mathscr{A}$ be an agent, ag is a tuple ($\mathscr{W}$, A_{QoS}, T) where :

- $\mathscr{W}$ represents the agent workload to be performed to accomplish a given task.
- A_{QoS} represents the set of QoS requirements which are defined as a tuple (rr, rc) where : (i) rr represents the required minimal memory and (ii) rc is the minimal required computing capacity.
- T: $\mathscr{W} \rightarrow A_{QoS}$ which assigns for the agent workload $w \in \mathscr{W}$ a set of QoS requirements in terms of RAM and CPU in A_{QoS} to be correctly executed.

For example, the agent r_7 within the MAS illustrated in Figure 1 requires at (t=t_0) (20 Mb) of CPU and (0.5 Gb) of RAM, whereas at (t=t_1) those requirements increase to (50 Mb) of CPU and (1 Gb) of RAM due to an increase in its performed workload.

Definition 5: (Multi-Agent System) A MAS is a tuple (A, E, I, O) (vowels decomposition [17]) where:

- A : represents the set of agents within the MAS.
- E : represents the virtual or real entity where the agents are embedded.
- $I \subseteq \mathscr{A} \times \mathscr{A}$ represents the set of direct interactions between the agents. An interaction denotes the exchange of messages between the agents;
- O : represents the set of organizations within the MAS.

For example, as shown in Figure 1, the MAS= (A, E, I, O) where : A= $(a_1, .., a_4, v_1, .., v_9, r_1, .., r_7)$; E= "The city"; I[1]= {(a1, v3),(a1, v1), (r3, r7)...}; O=(O_1, O_2, O_3) where O_1 is the group of rescuers, O_2 is the group of victims and O_3 is the group of ambulances.

3.3. Adaptive deployment of MAS into hybrid clouds

An adaptive deployment of MAS into a hybrid cloud consists of (i) allocating the appropriate public cloud resources once the existing ones are insufficient and/or (ii) redistributing the agents among the on/off premises once a load-balance issue occurs.

Let mas be the multi-agent system to be deployed in a hybrid cloud environment $\mathscr{H}_{\mathscr{C}}$. The deployment of mas into $\mathscr{H}_{\mathscr{C}}$ is given in definition 6.

Definition 6: Adaptive MAS Deployment The adaptive deployment of mas into $\mathscr{H}_{\mathscr{C}}$ is a function D: $\mathscr{A} \times \mathscr{W} \rightarrow C \times V_M$ which is invoked each time a load balance issue is detected due to the changes in agents workloads. It assigns for each agent $ag \in \mathscr{A}$ the suitable VM within the appropriate cloud $\mathscr{C} \in \mathscr{H}_{\mathscr{C}}$. We denote by $\mathscr{D} = \{(ag_1, D(ag_1, w_{ag_1})), (ag_2, D(ag_2, w_{ag_2})), \ldots, (ag_n, D(ag_n, w_{ag_n}))\}$ the set of agents and their allocated resources corresponding to the executed workload within each agent over-time.

The deployment of MAS on a hybrid cloud needs to be cost-efficient. In other words, the distribution of the agents across the on and off premise resources should result in minimal compute and data transfer costs. In the following section, we detail our proposed approach, which given a MAS and a hybrid cloud environment, dynamically provide a cost-efficient deployment.

[1] presented in Figure 1 with arrows

4. Proposed Approach to support MAS distribution on cloud environments

In this section we present an overview of the proposed framework to support MAS distribution and deployment on a given cloud environment (section 4.1). Afterwards, we describe in more detail the performance evaluation process to estimate and optimize the overall cloud deployment costs (section 4.2) as well as we present the extended Fiduccia-Mattheyses (E-FM) algorithm (section 4.3).

4.1. Architecture of the proposed framework to support MAS deployment on cloud environments

The proposed framework, shown in Figure 5, takes as input a given MAS and a given cloud environment (e.g. hybrid cloud, cloud federation) in order to achieve an adaptive cost-efficient deployment of the system across the underlying infrastructure. The framework incorporates two main processes: (i) ***a pre-deployment process***, and (ii) ***a deployment process***.

4.1.1. The pre-deployment process

The pre-deployment process represents a prediction phase to investigate and optimize the overall deployment costs of a given MAS on a given cloud environment. It is composed of two processes:

- The ***pre-selection process*** [2] takes as input a given MAS and a set of partitioning algorithms and provides as output a set of candidate algorithms that can be used to efficiently deploy the MAS across a distributed infrastructure. An efficient deployment of agent-based systems on distributed computational environments requires significant efforts to assign the agents to the corresponding VMs while minimizing the communication costs. To do so, various partitioning algorithms such as this of [5] [6] [18] have been used to distribute agents across different infrastructures. Based on a deep literature study, we proposed a conceptual framework [19] that defines a set of recurrent criteria related to (i) the MAS such as its type, the agents proprieties, the characteristics of the environment and the type of the interactions as well as; (ii) the target infrastructure. The framework allows analyzing the appropriateness of the partitioning algorithms for a given MAS type and giving guidelines to develop new

[2] The pre-selection process is out of the scope of this paper

distributed systems. As far as we know only the approach elaborated in [9] [10] focused on using the cloud environment to run distributed MAS. The proposed approach is dedicated to develop social simulations to be executed on a HPC cluster hosted in a single cloud environment. Further, the existing approaches for distributing applications on clouds [20] [4] [21] are too specific and dependent on the type of applications (more details in section 6.1). Therefore, we adapt and extend existing partitioning algorithms by taking into consideration the cloud specifications (compute costs, data transfer costs and latency, to name a few) in order to efficiently distribute MAS on different cloud environments (hybrid and multi-cloud environments). Afterwards, the conceptual framework will be enriched with additional extended algorithms that can be recommended to distribute a given MAS on a given cloud infrastructure.

- The ***performance evaluation process*** consists of estimating and optimizing the overall deployment costs by providing a suitable efficient distribution of the MAS across the cloud environment. It takes as input, the MAS to be deployed, the list of candidate algorithms provided by the pre-selection phase and the cloud specifications and provides as output an adaptive cost-efficient deployment solution of the MAS. The evaluation process is based mainly on using a reference machine to run a profiling phase of the MAS. It aims at collecting the required information including the execution time and the intra/inter communications within a MAS. The collected data is used as an input to the partitioning algorithms virtually simulated to distribute the system on the cloud resources. As shown in Figure 5, the prediction process is based on three essential services which are as follows: the *monitoring service*, the *decision making service* and the *invoking algorithms service*. Since the current work focuses on the prediction process, all the aforementioned services will be thoroughly described in the next section (4.2).

4.1.2. The deployment process

The deployment process takes as input the deployment solution provided by the pre-deployment process in order to assign each agent within the MAS to the corresponding cloud resource. In the next sections, we focus mainly on the performance prediction process in order to evaluate the E-FM algorithm and its

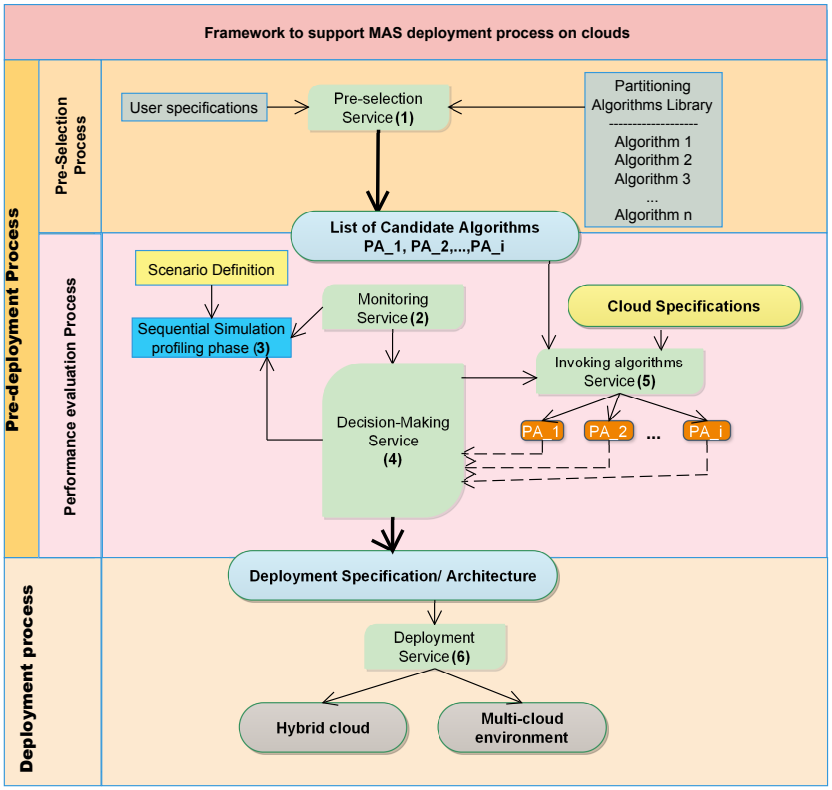


Figure 5: The proposed Framework to support MAS deployment process on a cloud environment

integral version. We assume that both of the algorithms are chosen as candidate algorithms from the pre-selection phase.

4.2. Performance evaluation process

The main steps of the performance evaluation process are presented in algorithm 1.

The process takes as input a MAS to be deployed, the hybrid cloud $\mathscr{H}_{\mathscr{C}}$, the number of steps required to finish the simulation (t_{end} is expressed in terms of time unit) and the list of the candidate partitioning algorithms L_{alg_p}. The main idea consists of running a profiling phase using a sequential simulation (non-distributed MAS) on a reference machine to perform many partitioning options and provide the suitable deployment solution. The algorithm starts by calculating the initial deployment by assigning each agent within the MAS to the corresponding cloud resource within $\mathscr{H}_{\mathscr{C}}$. Then, at each simulation step,

Algorithm 1 Pseudo-code of the Performance Evaluation Process

Input: MAS,$\mathscr{H}_{\mathscr{C}}$,$t_{end}$, L_{alg_p}
Output: $\mathscr{D}$

```
1: Calculate initial deployment
2: Start Sequential Simulation and trigger the Monitoring
3: step_sim ← 0
4: balance ← true
5: while step_sim ≠ t_end do
6:    Suspend the Simulation
7:    balance=Check the load-balance on used VMs
8:    if balance == false then
9:       D= Perform partition algorithms within L_alg_p
10:      Perform re-allocation according to D
11:   end if
12:   Resume Simulation
13:   step_sim ← step_sim + 1
14: end while
```

the CPU and RAM usage as well as the exchanged data between the agents residing on different VMs is analyzed to check the load-balance (line 2-7). If this is the case, the candidate algorithm within L_{alg_p} are separately invoked to restore the balance or allocate new resources if the existing ones are insufficient to fulfill the agents new requirements while maintaining reduced data transfer costs (line 9). Afterwards, the resources are allocated according to the new computed deployment solutions using the different algorithms and the simulation is resumed (Line 10-12). As shown in the algorithm 1, the process incorporates mainly three activities: (i) *the sequential simulation*, (ii) *the monitoring* and (iii) *load-balance and re-deployment*. The Table 1 summarizes all the notations used in this paper.

4.2.1. Sequential Simulation

The agent-based system is executed on a reference machine as a sequential monolithic application. At each simulation step, the collected data is logged and used to detect the need for invoking the partitioning algorithms with the aim of restoring the balance between the partitions allocating and/or de-allocating new cloud resources to fulfill the system requirements in terms of memory and computing capacities. The logged data consists of the detected execution time on the reference machine, the VMs used to deploy the agents, the size of the intra/inter exchanged data between the agents. The data is gathered using a monitoring service, which is triggered at the beginning of the simulation. Then,

it is communicated to the decision-making service (Figure 5), which is responsible for invoking all the candidate algorithms to assess different partitioning options. The agent-based system is not affected by any of the partitioning mechanisms.

4.2.2. Monitoring

Due to the complex behaviors of the simulated entities and their interactions, the computational structure of the simulation may change during run-time, which leads to an unbalanced system and affects the overall performance.

Symbols	Definitions
$\mathrm{ag_i}$	agent i / (i=1,..., n)
$\mathscr{A}$	the set of agents
$\mathscr{H}_{\mathscr{C}}$	the hybrid cloud
$\mathscr{I}_p$	the on-premise infrastructure
$\mathscr{C}_p$	the set of public cloud providers
$\mathrm{T_{Unit}}$	time unit defined by the cloud provider to apply charges.
$\mathrm{D_{Unit}}$	data unit defined by the cloud provider to apply charges.
$Cost_{Exec_{T_{unit}}}$	cost associated to $\mathrm{T_{Unit}}$.
$Cost_{Exec_{D_{unit}}}$	cost associated to $\mathrm{D_{Unit}}$.
β	bandwidth
λ	communication Latency for a given Cloud
PI	performance Index between the source machine and the target one.
$\mathrm{T}_{exec_{S_{VM_k}}}$	execution time of VM_k on the source machine performing the sequential simulation.
$\mathrm{T}_{exec_{T_{VM_k}}}$	estimated Execution Time of VM_k on the target machine performing the sequential simulation.
T_{exec_T}	The execution time of one iteration of the MAS.

Table 1: Notations.

Therefore, computing time and agents loads including computational and communication loads are monitored at the beginning of each iteration. The design of an agent always requires a deep specification of the environment. An environment within a MAS can have the following characteristics [22]:

- Fully observable (vs partially observable): The agent has access to the complete state of the environment at each point of time.
- Deterministic (vs Stochastic) : The next state of the environment is completely determined by the current state and the action executed by the agent.

- Episodic (vs sequential) : An agent action is divided into atomic episodes. Decisions do not depend on previous decisions/actions.
- Static (vs dynamic) : The environment is unchanged while an agent is deliberating.
- Discrete (vs continuous): A limited number of distinct, clearly defined percepts and actions.
- Single agent (vs multi-agent): An agent operating by itself in an environment.

We are interested in a MAS, in which the environment has the following characteristics: fully observable, deterministic, sequential, dynamic, discrete and multi-agent. For our case study of evacuation described in section 2, the aforementioned proprieties characterize our simulation environment. Therefore based on the history of the previous iteration, we can determine for the current iteration the load that will be executed by each agent based on the pre-achieved behavior. Consequently, we can determine the time that will be taken by each partition to execute the overall workload. The estimated execution time on a used VM within the hybrid cloud $\mathscr{HC}$ during a simulation step is determined using the captured execution time on the reference machine. Indeed, each agent has a workload to be executed, which can be expressed in terms of the number of performed instructions. Further, the reference machine is characterized by its MIPS (Millions of Instructions Per Second).

Thus, let $\{ag_i \mid i = 1..n\} \subseteq \mathscr{A}$ be a set of agents deployed on the VM_k within the hybrid cloud $\mathscr{HC}$. Each agent $ag_i \in \{ag_i \mid i = 1..n\}$ has a set of instructions N_{ag_i} to be performed. Therefore the execution time of $\{ag_i \mid i = 1..n\}$ on the reference machine is determined in equation 1 as follows:

$$T_{exec_{S_{vm_k}}} = \frac{\sum_{ag_i \in VM_k} N_{ag_i}}{MIPS * 10^6} \tag{1}$$

The numerator corresponds to the total number of instructions performed by the agents deployed on the virtual machine VM_k within $\mathscr{HC}$. The calculated execution time on the source machine is used to determine the estimated execution time on the target virtual machine VM_k. Thus the estimated execution time is determined using the equation 2:

$$T_{exec_{T_{VM_k}}} = T_{exec_{S_{vm_k}}} * PI \tag{2}$$

The first element of the equation denotes the captured execution time on the reference machine multiplied by the Performance Indicator (PI) that gives the

ability of the target machine to execute the load compared to the source machine.The estimated execution time of one iteration corresponds to the maximal execution time recorded over all the used VMs within $\mathscr{H}_{\mathscr{C}}$. The recorded time value, shown in equation 3, is used to detect the imbalance between partitions if it exists and invoke the partitioning algorithm of the load-balance mechanism.

$$T_{exec_T} = max(T_{exec_{T_{vm_k}}}) \tag{3}$$

The load-balance mechanism is activated once the execution time on the target infrastructure exceeds a given threshold value [3].

4.2.3. Load-balance

Once an imbalance is detected, the candidate algorithms are invoked in parallel to start the partitioning phase. In fact, each instance of the running algorithms uses the logged simulation data as well as the hybrid cloud specifications (cloud provider, charges, latency, bandwidth and virtual machines capabilities to name a few) to proceed with the virtual partitioning. The number of the executed instances depends on the number of the candidate algorithms. At the end of the partitioning phase, each running instance generates the total cost of the deployment solutions provided using the partitioning algorithms. All the costs are logged in order to be used at the end of the simulation to evaluate the overall performance of all the algorithms across the hybrid cloud.

A generated cost incorporates three metrics namely, the execution time cost, the communication cost, and the migration cost defined in a previous work[19]. The performance evaluation process is used to compare the performance of a new adapted algorithm called **E-FM** to its original **FM** version .

4.3. Candidate Partitioning Algorithms

In this section, we highlight the graph-based partitioning problem and present our extended version of the FM algorithm used to find an efficient deployment.

4.3.1. Problem Statement

We map the task of finding a cost-efficient deployment to a graph-partitioning problem. Graph partitioning is NP-hard problem [23]. It consists of dividing a given graph into well balanced K sub-graphs while minimizing the number of

[3]The threshold value is determined during the experimentation

the interconnecting edges. The selection of the graph approach is not arbitrary. For all the existing partitioning algorithms (discussed in more details in section 6) for agent-based systems, only graph-based partitioning algorithms consider explicitly the minimization of the communication costs, which is a key issue within distributed environments. Our objective is to assign each agent $ag \in \mathscr{A}$ within a MAS to one of the VMs within the hybrid cloud $\mathscr{H}_{\mathscr{C}}$. Further, we aim at finding a deployment solution, in which the overall costs are minimized.

Let G=($\mathscr{A}$,$\mathscr{E}$,W_{ag},W_e) be an un-directed graph where :

- $\mathscr{A}$= (ag_1, ag_2,.., ag_i, $i \geq 1$) is the set of agents within a given MAS to be deployed on cloud resources (Definition 4).
- $\mathscr{E}$ represents the set of edges connecting the agents. Within a MAS an edge denotes an interaction between two agents (Definition 5).
- W_{ag} denotes a set of requirements assigned to each agent ag_i. Each agent has a list of needs in terms of computing and memory resources.
- W_e represents an edge' s weight. It denotes the communication overhead between the agents. C=($w_{i,j}$) is the adjacency matrix of the graph G. If there exists a communication between ag_i and ag_j, $w_{i,j}$ represents the amount of the communicated data.

Let M =(m_{kl}) be a matrix where m_{kl} depicts the cost of transferring data ($/Gb) between the virtual machines k and l within the hybrid cloud $\mathscr{H}_{\mathscr{C}}$. If both virtual machines belong to the same cloud provider [4], then the data exchange cost m_{kl} =0. Otherwise, m_{kl} represents the inter-cloud bandwidth utilization cost.

More formally, for each agent $ag_i \in \mathscr{A}$ within the MAS we consider the following decision variable:

Y_{ijk} equal to 1 if ag_i is deployed on the virtual machine VM_k within the cloud C_j, and 0 otherwise.

The main objective is to minimize the inter-cloud communication, consequently the sum of the edges weight between the VMs belonging to private and public clouds. Our goal is to find two partitions of the graph G where the first and second partitions present the private and public clouds respectively. Thus, we use the variable p_{ij} which is equal to 1 if the virtual machines VM_i and VM_j

[4]means both of machines are on the private cloud or within the same public cloud provider

are in different clouds and, 0 otherwise. Therefore, we consider the following objective function depicting the sum of the inter-edges communications (F) to be minimized (equation 4):

$$\mathscr{F} = \sum_{i,j}^{A} \sum_{k,l}^{\mathscr{C}} \sum_{t,n}^{VM} (Y_{ikt} Y_{jln} P_{tn} m_{tn} w_{ij}). \tag{4}$$

Also more constraints represented in the equations 5 and 6 are required to describe properly our problem:

$$\forall j \in \mathscr{C}, k \in VM, \sum_{i}^{A} (Y_{ijk} rc) \leqslant mc Y_{ijk}. \tag{5}$$

$$\forall j \in \mathscr{C}, k \in VM, \sum_{i}^{A} (Y_{ijk} rr) \leqslant mr Y_{ijk}. \tag{6}$$

Equations 5 and 6 indicate that the total requirements expressed in terms of CPU and RAM of a set of agents residing in the same machine, should not exceed the maximum capacity of that machine.

4.3.2. Extended-Fiduccia-Mattheyses

To partition the graph and find a cost-efficient deployment, we extend the FM algorithm, which was firstly introduced in [24]. FM is a bi-partitioning algorithm that takes as input a graph and provides as output a graph partitioned into two subsets such that the number of the edges interconnecting the partitions is minimized. Our extended version of FM is different in the sense that:

- We are interested in minimizing the cost of the transferred data between the on and off premise resources instead of minimizing the number of communications (number of the inter-edges in the graph). In fact, the number of communications may be high, whereas the amount of the exchanged data is low and vice versa.
- Further, we want to find public cloud VMs that fulfill the agents requirements in terms of memory and computing resources. Thus, we may find public cloud VMs with minimal costs, however, its capabilities are insufficient to deploy the agents.

E-FM algorithm (Algorithm 2) takes as input the MAS and the hybrid cloud environment. It provides as output a deployment solution $\mathscr{D}$ (Definition 6) that consists of assigning each agent within the MAS to a pair: the cloud provider and a VM in the hybrid cloud. The algorithm is performed on two phases: (i) ***the***

first phase allows generating a deployment graph on which the partitioning will be performed (Line 1). ***The second phase*** denotes the re-partitioning of the graph in order to meet the agents new requirements and minimize the communication overhead (Line 2-Line 19).

Resource Allocation

The first phase consists of assigning the agents within the MAS to the corresponding resources within the hybrid cloud. The objective is to generate a graph on which the partitioning will be performed. The graph depicts the communication costs between the **on** and **off premise VMs**, where the agents are deployed. An example of the graph is shown in Figure 6. The generated graph consists of four layers. The first layer (the smallest nodes) represents the agents within the MAS and the communications between them. The second layer shows the VMs, which are connected if they are allocated to interacting agents. The third layer shows the cloud providers, where the VMs are hosted. The forth layer, represents the private and public infrastructures, to which the cloud providers belong. The graph edges are annotated with weights that are defined as the data exchange between the agents within the MAS multiplied by the bandwidth price in the hybrid cloud.

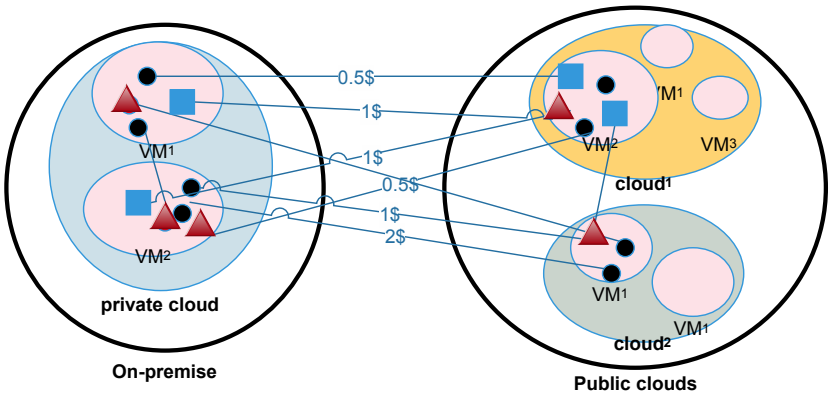


Figure 6: Generated deployment nested graph

In order to generate the graph, we need to find an initial resource allocation. To do so, many algorithms can be used such as the first fit algorithm, the best fit algorithm and random allocation, to name a few. The impact of an algorithm used for initial allocation is out of the scope of this paper. In the frame of this work, we used the first fit algorithm that consists of assigning the agents within the MAS and leasing low cost public cloud VMs. It takes as input, the list of

the agents, as well as the list of the clouds and VMs within the hybrid cloud. As output, it provides a deployment solution that assigns each agent to the first available VM that fulfills the agent's needs in terms of CPU and RAM.

The partitioning phase This phase consists of re-partitioning the generated graph in order to minimize the communication costs between the on and off premise resources. The partitioning is performed at the fourth level of the

Algorithm 2 Pseudo-code E-FM

Input: MAS, $\mathscr{H}_{\mathscr{C}}$
Output: $\mathscr{D}$

```
1: Resource allocation: Generate a deployment graph
2: repeat
3:    Compute the gain for each agent
4:    Order agents into a bucket
5:    repeat
6:       Select the agent with Maximum gain value
7:       if agent needs are verified then
8:          Move agent to the opposite side
9:          Lock the Moved agent
10:         Delete the agent from the bucket
11:         Update the gain for the agent neighbors
12:         Compute the cumulative gain
13:      else
14:         Select next maximum gain in the bucket
15:         Goto to instruction 7
16:      end if
17:   until No free cells
18:   Update partitions: Confirm swaps
19: until no reduction in cut-size
20: Return partitions
```

generated graph, which shows the distribution of the agents between the on-premise infrastructure and the set of the public cloud providers. During this phase, E-FM redistributes the agents so that the updated partitions represent the lowest communication costs. The agents migrate from and to the on-premise resources. The decision of migrating an agent is based on computing a gain value determined in equation 7. Let $ag_i \in \mathscr{A}$ be an agent deployed on a VM within $\mathscr{H}_{\mathscr{C}}$ and g_i is the gain associated to the agent ag_i to be migrated from/to the on-premise infrastructure.

$$g_i = g_{i,Exec_T} + g_{i,Com_D} \tag{7}$$

Where $g_{i,Exec_T}$ is the gain associated to the agent ag_i in terms of the data

transmission time, which affects the overall execution time. Whereas, g_{i,Com_D} is the gain associated to the agent ag_i in terms of the amount of the exchanged data.

In our extended algorithm we distinguish two cases for determining the agent gain value:

- If the agent is deployed on a VM hosted within the on-premise infrastructure, its gain is defined as the sum of both equations 8 and 9. Let $ag_i \in \mathscr{A}$ be an agent deployed on a $VM_j \in \mathscr{I}_p$, $g_{i,Exec_T}$ and g_{i,Com_D} are the gains associated to the agent ag_i, to be migrated from $\mathscr{I}_p$ to $\mathscr{C}_p$, in terms of data transmission time and size of the communicated data respectively.

$$g_{i,Exec_T} = (\frac{(\frac{E_S(ag_i,\mathscr{C}_p)-I_F(ag_i,\mathscr{I}_p)}{\beta}) + \lambda}{T_{unit}}) * cost_{Exec_T} \tag{8}$$

$$g_{i,Com_D} = (\frac{E_S(ag_i,\mathscr{C}_p) - I_F(ag_i,\mathscr{I}_p)}{D_{unit}}) * cost_{Com_D} \tag{9}$$

  Where within both equations 8 and 9, $E_S(ag_i,\mathscr{C}_p)$ is a function that returns the size of the data received by the agent ag_i and sent from a set of agents deployed on VMs hosted within $\mathscr{C}_p$. Whereas, $I_F(ag_i,\mathscr{I}_p)$ is a function that returns the size of data sent by ag_i to a set of agents deployed on VMs hosted within $\mathscr{I}_p$.

- If the agent is deployed on a VM hosted in one of the public cloud providers, its gain is defined as the sum of both equations 10 and 11. Let $ag_i \in \mathscr{A}$ be an agent deployed on a $VM_j \in \mathscr{C}_p$, $g_{i,Exec_T}$ and g_{i,Com_D} are the gains associated to the agent ag_i, to be migrated from $\mathscr{C}_p$ to $\mathscr{I}_p$, in terms of data transmission time and size of the communicated data respectively.

$$g_{i,Exec_T} = (\frac{(\frac{E_F(ag_i,\mathscr{I}_p)-I_S(ag_i,\mathscr{C}_p)}{\beta}) + \lambda}{T_{unit}}) * cost_{Exec_T} \tag{10}$$

$$g_{i,Com_D} = (\frac{E_F(ag_i,\mathscr{I}_p) - I_S(ag_i,\mathscr{C}_p)}{D_{unit}}) * cost_{Com_D} \tag{11}$$

  Where within both equations 10 and 11, $E_F(ag_i,\mathscr{I}_p)$ is a function that returns the size of the data sent from the agent ag_i and received by a set of agents deployed on VMs hosted within $\mathscr{I}_p$. Whereas, $I_S(ag_i,\mathscr{I}_p)$ is a function that returns the size of data sent by a set of agents deployed on

VMs hosted within $\mathscr{C}_p$ and received by ag_i.

In both equations 8 and 10, the gain in the transmission time is multiplied by the computing costs. Whereas, in the equations 9 and 11, the gain in the amount of the transferred data is multiplied by the bandwidth price within the hybrid cloud.

Once the gains are computed and ordered (Line 3-Line 4), the migration of the agent with the maximum gain is only valid if the hosting infrastructure contains a VM that fulfills the agent needs in terms of CPU and RAM (Line 5-Line 17). The algorithm stops when no improvement in the inter communication costs is recorded in two successive iterations.

5. Experiments and Results

The proposed approach is implemented using the JADE agent platform [5], which is compliant with the FIPA [6] specifications. In other words, the performance evaluation process is implemented as an agent-based system where each of the services described above (in section 4.2) is performed by a single agent. Furthermore the graph-based algorithms FM and E-FM are implemented using the Java language and Eclipse. We conducted a set of experiments to compare both algorithms using different hybrid cloud settings and various simulation scenarios. As input, we considered five different configurations to run different simulation scenarios of the MAS presented in section 2. Each configuration in Table 2 consists of the numbers of the victims, rescuers and ambulances. The configurations vary in terms of the total number of agents within the simulation.

The different hybrid cloud settings have been simulated. Indeed, many research works [25] [26] [27] focused on using the simulation technique for cloud environment testing. Indeed, it allows decreasing the complexities of dealing with the underlying infrastructure while focusing on the quality concerns. The simulation of cloud system has shown its usefulness in: (i) investigating the effectiveness of various approaches dedicated for provisioning, deployment and scheduling; (ii) offering a controllable environment where researchers can test their approaches in a repeatable way without paying any cloud fees and (iii)

[5] http://jade.tilab.com/
[6] http://www.fipa.org/

Configurations (Total agents number)	Victims Number	Rescuers Number	Ambulances Number
200	130	40	30
400	270	100	30
600	400	120	80
800	500	200	100
1000	600	250	150

Table 2: MAS Configurations.

providing reasonable and near results compared to the real conducted experiments such as in [27] [28]. Thus, in the current work, the simulation of the hybrid cloud environment exempted us from dealing with the complexities of the underlying infrastructure while ensuring reasonable simulated results. As shown in Table 3, the used hybrid cloud interconnects an on-premise infrastructure with two VMs and two public cloud providers (P_1 and P_2). Each cloud provider offers 30 types of VMs, located in two different availability zones (see Table 3) . Both private and public infrastructures are interconnected through a bandwidth, which we varied its values to determine its impact on the overall deployment costs.

Table 3: Hybrid cloud settings

<table>
<tr><th colspan="2"></th><th>Hybrid setting 1</th><th>Hybrid setting 2</th><th>Bandwidth</th></tr>
<tr><td>On-premise</td><td>Vms number</td><td colspan="2">2</td><td rowspan="4">B1= 100 Mb/s
B2=300 Mb/s</td></tr>
<tr><td rowspan="3">public cloud</td><td>providers</td><td>P1</td><td>P2</td></tr>
<tr><td>VM types number</td><td colspan="2">30</td></tr>
<tr><td>Availability zones</td><td colspan="2">R1: eu-central-1
R2: ap-southeast-1</td></tr>
</table>

The experiments were conducted on a laptop with a 64-bit Intel Core 2.10 GHz CPU, 4 GB RAM and Windows 7 as operating system. Further, the experiments were performed under the following assumptions:

- All the messages have the same size. We used the known maximum message size value, to overestimate the size of the rest of the messages. Our objective is to properly provide an accurate deployment with minimum communication costs. Thus using the overestimation whether the agents are exchanging small or large data size will improve the partitioning re-

sults.

- The latency is fixed for each used availability zone in hybrid cloud settings. It is represented as the mean of 10 measurements for each zone.

5.1. Experiment 1

In this section, we compare E-FM to FM using the following efficiency criteria:

- The inter-data transfer costs.
- The overall deployment costs.

The partitioning algorithms are virtually simulated to provide deployment solutions on the hybrid cloud setting 1 shown in Table 3 using as availability zone R_1 and 100 Mbit/S of bandwidth.

Figure 7 presents the comparison between the obtained deployment solutions generated using E-FM and FM in terms of inter transferred data within the hybrid cloud. For all the configurations (see Table 2), our extended version of FM provides better results in terms of reduced communication costs between the on/off premise resources. Thus considering the minimization of the data transfer costs is more pertinent than focusing only on reducing the number of the communications (number of edges interconnecting the partitions). As a consequence, the deployment solution provided by FM results in increased inter-communication costs. Further, as shown in Figure 8, E-FM provides the lowest overall deployment solution costs compared to FM. Due to the large amount of transferred data within the FM solution, the time of processing and executing the simulation are both increased, consequently the deployment costs present highest values compared to those found using E-FM.

5.2. Experiment 2

In this section, we present the impact of varying parameters related to the hybrid cloud setting on the data transfer and overall deployment costs. E-FM outperforms FM within the same hybrid cloud and provides a deployment solution with reduced costs. However, the extended algorithm is based mainly on the computation of gains in terms of data transfer costs through a formula that uses provider cost model, bandwidth and network latency. Thus, varying those parameters may have a huge impact on the system performance as well as the overall deployment costs. The experiments are conducted using the E-FM partitioning algorithm while varying at each time one of the following factors:

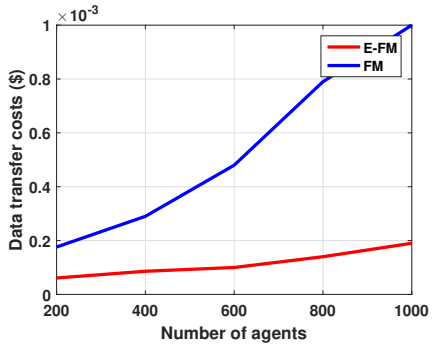


Figure 7: Inter Data Transfer costs of the deployment solutions on the hybrid cloud setting 1 using FM and E-FM

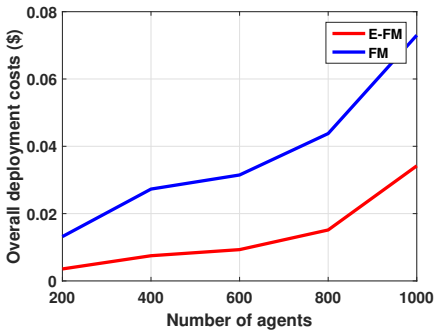


Figure 8: Overall costs of the deployment solutions on the hybrid cloud setting 1 using FM and E-FM

- Varying the cost model through the use of different public cloud providers (P_1, P_2)
- Varying the network latency by using different availability zones (R_1, R_2) within the used hybrid cloud.
- Varying the bandwidth interconnecting the on/off premise infrastructures (B_1, B_2).

5.2.1. Impact of public cloud providers

In this section, we present the impact of varying the used public cloud providers within the hybrid infrastructure while maintaining the same bandwidth and availability zone. The cloud market is in continuous growth with competitive third parties, therefore choosing the right public provider is a challenging issue. We consider the use of two virtual machines vm_1 and vm_2 within

the cloud providers P_1 and P_2 respectively. Both of the VMs are similar in terms of technical capabilities. As shown in the Figure 9, with the use of the cloud provider P_2, E-FM gives reduced overall deployment costs compared to provider P_1. The results are explained by the fact that each cloud provider has its own cost model, which impacts the overall deployment costs.

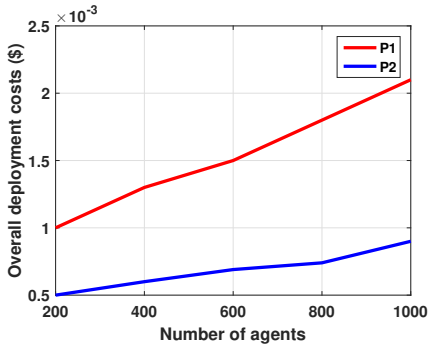


Figure 9: Overall costs of the deployment solution on the hybrid cloud setting 1 and 2 with E-FM

5.2.2. Impact of availability zones

In this section, we present the impact of the used availability zones within the same cloud provider, while maintaining a fixed bandwidth. Indeed, the data centers within an IaaS provider are distributed in different locations in the world. Those locations contain independent regions. Each region is composed of a set of isolated interconnected availability zones. The latency within a hybrid cloud architecture is a critical issue, however with the use of different regions and availability zones, the latency problem is more complicated and may affect the system performance and induce extra costs. Thus, studying the impact of the network latency within a hybrid architecture is of great importance to provide a cost-efficient deployment solution.

The experiments are performed using the hybrid cloud setting 2 (see Table 3) by varying both the region and availability zone of the allocated Vms within P_2. Figure 10, shows the overall costs of the deployment solutions provided by E-FM using two different availability zones within two different regions.

The results show that the network latency induced by using different availability zones within different regions has a huge impact on the overall deployment costs. As shown in Figure10, the extended algorithm used with the region R_1 provides better results (reduced deployment costs) compared to the selection

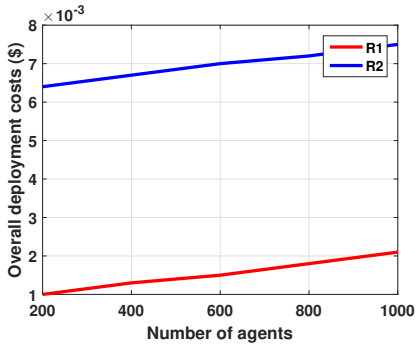


Figure 10: Overall costs of the deployment solution on the hybrid cloud setting 1 using R1 and R2

of the region R_2. The results are explained by the fact that the induced latency within R_2 is large and increased with the amount of communicated data between the on/off premise VMs. To minimize the impact of the network latency, it is important to select the right locations to allocate the resources.

5.2.3. Impact of bandwidth

In this section we present the impact of the used bandwidth within the hybrid cloud on the data transfer costs. The experiments are conducted using the hybrid cloud setting 2 with R_2 as an availability zone.

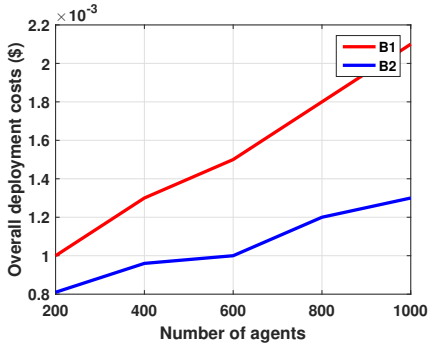


Figure 11: Overall deployment costs within the hybrid cloud setting 2 using B_1 and B_2

As shown in Figure 11, the used bandwidth has an impact on the deployment costs for all the configurations. E-FM provides better results (reduced costs) by using B_2 within the hybrid cloud. The allocation of the right bandwidth, in the context of distributed environments is of great importance. On one hand the use of a small bandwidth may cause extra delays and consequently lead

to undetermined execution time, impact the system performance and inhibit the efficiency of the used cloud resources. On the other hand, allocating large bandwidth may result in high deployment costs.

Thus given the various parameters that may impact the system performance and increase the overall costs, an offline approach to estimate and optimize the deployment of MAS to hybrid clouds is of great importance.

6. Related Work

In this section, we provide an overview of existing prediction and performance evaluation approaches as well as the partitioning strategies of MAS on distributed environments.

6.1. Prediction-based approaches for cost-efficient deployment on cloud environments

Estimating and optimizing cloud deployment costs have been widely explored in several research work with various types of applications and diverse cloud environments. Table 4 provides a synthesis of a selection of existing prediction-based approaches according to the following criteria: (i) Focus; (ii) Application type; (iii) Granularity; (iv) Metrics; (v) Cloud environment.

In [20], the authors present the CloudWard Bound for partially outsourcing enterprise services to a hybrid cloud environment. Based on the application performance requirements as well the privacy restrictions a set of candidate components are selected to be migrated to the public cloud. The authors use a set of metrics including the amount of executed workload, the storage capacity and the transaction delays to provide an efficient hybrid cloud deployment. In [29], the authors present both a methodology and chain tool for cloud benchmarking and cost-efficient outsource of a set applications including databases, file systems and JVM applications to the cloud. The proposed approach assumes the use of a single VM within a cloud provider. In [30], the authors are interested in achieving a cost-efficient deployment of deadline constrained bag-of-tasks applications to a multi-vendor hybrid cloud. A set of scheduling algorithms have been studied to predict and optimize the tasks outsource cost. In [31] a prediction approach to ensure effective VM-provisioning and admission control of multi-tiers web-applications is proposed. The approach provides an adaptive automatic deployment solution on a given IaaS provider (Amazon EC2) by using a set of auto-scaling and admission control algorithms. In [32], a cloud

broker system called BioCloud is introduced. Given the budget constraints and the workflow to be executed, BioCloud achieves a cost-efficient deployment in a multi-cloud environment. It incorporates a profiler that estimates and optimizes the overall costs before proceeding with the real deployment. To the best our knowledge, we are the first to address the prediction and optimization of deployment costs for MAS on cloud environments.

In the aforementioned work, each proposed prediction-based approach is dedicated to be used with a given type of application and a given cloud environment. Each approach implements a set of scheduling and distribution algorithms to generate different deployment solutions and defines objective functions to be maximized or minimized based on the target aim. Granularity, which represents the considered level of detail varies from one type of application to another. In MAS, we consider the *agent* as the deeper level of detail in the deployment process. Compared to the defined granularities in [20] [29] [30] [31] [31] an agent is an active autonomous and continuously running entity that can be goal-oriented, mobile, communicative and flexible, to name a few. Such characteristics make the MAS different from the rest of the above applications types in the sense that:

- A MAS, with autonomous mobile agents that can migrate (move) from one VM to another multiple times during the simulation, requires a continuous monitoring of its state over-time to detect any issue (imbalance and/or insufficient resources). Whereas, within the rest of the applications the deployed entities (tasks, jobs, components, etc) are immobile and the need for additional resources is triggered by external events such as the large numbers of user requests, tasks or jobs to be processed.

- Based on the simulated system, an agent may have a set of requirements such as CPU, RAM, bandwidth, security and availability, to name a few. Such requirements may change continuously over-time and new resource allocation may be required to fulfill the ever-changing needs. In the existing approaches, only few QoS requirements are taken into consideration specifically CPU and RAM loads. While, additional requirements induce more challenges in finding the right cloud resources.

Further, since MAS can be used to model and simulate different system types, our prediction-based approach is generic provided that a mapping from the considered application to a MAS can be achieved.

Moreover, additional research work have been elaborated to predict the performance of simulations and specifically MAS. In [33], the authors focus on predicting the performance of parallel synchronous simulations mainly the simulation of computer networks. A model using sequential simulation as input and taking into consideration several factors including load-balance, communication overhead, computation granularity and partitioning was developed. The authors show the efficiency and the correctness of their model by comparing the predicted performance with the real system performance. In [34], performance analyzer tools were developed to predict the performance of parallel discrete event simulations(asynchronous and synchronous models) before proceeding with the real implementation. The developed tools predict the speedup change of the simulation model with the number of the used processors. In [18], an operational framework for evaluating partitioning mechanisms for agent-based simulations on a cluster is presented. The authors used a sequential simulation to test multiple partitioning mechanisms and recommend a suitable one for a given initial configuration. Similarly to the aforementioned works, we consider the use of sequential simulation running as a monolithic application on a reference machine in order to analyze and profile its execution over time. However, our goal is to provide an adaptive cost-efficient deployment of agent-based systems on cloud environments specifically hybrid ones. Consequently, additional factors including cloud market, network state (latency, bandwidth), characteristics of the VMs and monetary costs need to be considered to properly distribute the system and enhance its performance. Therefore the key novelty of the proposed performance evaluation process is the incorporation of both hybrid cloud specifications and MAS performance factors to provide a proper deployment solution that assigns each agent to the corresponding cloud VM while maintaining reduced communication costs.

6.2. Partitioning mechanisms for MAS

The distribution of agent-based systems across distributed environments has been addressed in many research work [5] [6] [18]. The existing distribution approaches consist mainly of partitioning the agents between the different resources composing the used infrastructure. The partitioning approaches applied to MAS fall into the following three categories: (i) **Cluster-based approach** [6] [35] [36] in which the agents are clustered based on given criteria (membership, position, etc) and then assigned to different machines; (ii) **Grid-based approach** [37] [38], called also region-based approach, consists of decompos-

ing the space component (environment) into multiple portions. Each portion together with the set of agents residing on it are assigned to a machine; (iii) **Graph-based approach** [39] [40] consists of representing the agents network in the form of a graph which can be divided into sub-graphs (partitions). Then each partition is assigned to a machine while minimizing the communication overhead. In our work, we used a graph-based algorithm to provide a cost-efficient deployment of a MAS on a given hybrid cloud. Among all the existing partitioning approaches for agent-based systems, only graph-based partitioning algorithms consider explicitly the minimization of the communication costs, which is a key issue within distributed environments specifically within cloud infrastructures.

Graph partitioning has been applied to different application types. In [40], the authors present a graph-based Automatic Load Balancing (ADLB) strategy developed with the aim of distributing Agent Based Models on High Performance Computing (HPC) infrastructure. The proposed strategy uses a monitoring approach to detect the imbalance and an activation mechanism to invoke the load-balance. Likewise, we used the computation time per iteration to detect the unsteadiness between the partitions. Otherwise, distributing ABS across hybrid cloud rises more challenges to consider in order to meet all the performance requirements during run-time. Consequently, a partitioning algorithm should incorporate additional factors such as monetary costs, size of the communicated data, latency, bandwidth and cloud charges while restoring the balance between the partitions.

Also in [39], the authors propose a graph-based dynamic load-balancing mechanism for parallel discrete-event agent-based large-scale nanoscopic traffic simulation. The elaborated approach consists of representing the spatial network as a weighted graph using workload and communication. To do the load balance, the authors fixed a threshold that invokes the algorithm to do the repartitioning each time it is needed. Likewise, we focus on using weighted graphs using the workload and the communication captured during the sequential simulation. Otherwise, we do not consider the geographical aspect to determine the agents migration. In our work the clustering of the agents is more related to the frequency of their inter and intra communications rather than their geographical positions which do not necessarily minimize the inter-communication overhead between partitions. In [41] , the authors discuss the challenges related to a carpooling agent-based application including the matching problem and the partitioning to have systems that are more scalable. Since the agents

may rapidly change their preferences about their partners, a frequent matching is needed to meet the requirements of the application. However, due to its complexity, the matching problem is partitioned into sub problems executed in parallel. The authors used graph partitioning in order to have disconnected sub-graphs (minimize connectivity between agents sets) and equal sized partitions. The normalized cut is used to achieve an efficient partitioning. Proof of concept experiments were conducted with a small number of agents (15 agents). In our work we consider the distribution of large-scale MAS (thousands) on a hybrid cloud infrastructure where the partitioning problem is proven to be more complicated and requires considering different metrics with the aim of ensuring an efficient distribution at run-time.

7. Conclusions and Future work

We proposed a prediction process to estimate and optimize the deployment costs of a given MAS on a given hybrid cloud environment. Moreover, we extended the FM algorithm to provide an adaptive cost-efficient deployment solution. The former process consists of simulating virtually various partitioning options and assessing their relative effects on the system performance by varying both scenarios definitions and hybrid cloud specifications. Our implemented approach highlights the efficiency of our extended algorithm E-FM compared to its original version and shows that the deployment costs are sensitive to multiple factors such as the simulated scenarios and hybrid cloud specifications including the public cloud providers and their cost models, the bandwidth and the network latency.

As a future work, we intend to improve our extended algorithm E-FM to be performed on k partitions ($K > 2$). E-FM is a bi-partitioning algorithm used to minimize the communication costs between the on/off premise resources. However, in the case where the hybrid cloud is composed of more than one public cloud provider, we may end up with high deployment costs even if the communication costs between the on/off resources is reduced. Indeed, the high costs may be due to an excessive data transfer across the used public cloud providers, which also needs to be kept minimized to optimize the deployment. Moreover, to ensure the genericity of our prediction-approach we intend to use the MAS as a finality to model and simulate different types of applications. Consequently, we provide a broker system that can provide automatic adaptive deployment solutions regardless the application type. Moreover, we intend to

explore more partitioning algorithms to extend the selection service partitioning library and enhance the generated deployment solution. Further, in order to prove the efficiency and reliability of our prediction approach, we intend to perform real deployment of MAS on a hybrid cloud environment and compare the results with those already estimated.

References

[1] N. Bellamine-Ben Saoud, T. Ben Mena, J. Dugdale, B. Pavard, M. Ben Ahmed, Assessing large scale emergency rescue plans: an agent based Approach, International Journal of Intelligent Control and Systems 11 (4) (2006) 260–271, special Issue on Emergency Management Systems. URL https://hal.inria.fr/hal-00952213

[2] M. W.Macy, R. Willer, From factors to actors: Computational sociology and agent-based modeling, Annual Review of Sociology 28 (2002) 66–143.

[3] J. Q. Niu, K. Zhang, H. Zheng, An agent-based modeling approach at molecular scale for biochemical networks: Simulating from stochastic molecular events, in: 2009 3rd International Conference on Bioinformatics and Biomedical Engineering, 2009, pp. 1–4. doi:10.1109/ICBBE.2009.5162306.

[4] N. Kaviani, E. Wohlstadter, R. Lea, Partitioning of web applications for hybrid cloud deployment, Journal of Internet Services and Applications 5: 14 (2014) 1–17. doi:doi:10.1186/s13174-014-0014-0.

[5] G. Vigueras, M. Lozano, J. Orduna, F. Grimaldo, A comparative study of partitioning methods for crowd simulations, Applied Soft Computing 10 (1) (2010) 225 – 235. doi:http://dx.doi.org/10.1016/j.asoc.2009.07.004.

[6] Y. Wang, M. Lees, W. Cai, S. Zhou, Nanyang, M. Y. H. Low, Cluster based partitioning for agent-based crowd simulations, in: Winter Simulation Conference, WSC 09, Winter Simulation Conference, 2009, pp. 1047–1058.

[7] G. Theodoropoulos, Y. Zhang, D. Chen, R. Minson, S. J. Turner, W. Cai, Y. Xie, B. Logan, Large scale distributed simulation on the grid, in: Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on, Vol. 2, 2006, pp. 63–63. doi:10.1109/CCGRID.2006.1630953.

[8] D. Chen, G. K. Theodoropoulos, S. J. Turner, W. Cai, R. Minson, Y. Zhang, Large scale agent-based simulation on the grid, Future Generation Computer Systems 24 (7) (2008) 658 – 671. doi:https://doi.org/10.1016/j.future.2008.01.004.
URL http://www.sciencedirect.com/science/article/pii/S0167739X08000058

[9] P. Wittek, X. Rubio-Campillo, Scalable agent-based modelling with cloud hpc resources for social simulations, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, 2012, pp. 355–362. doi:10.1109/CloudCom.2012.6427498.

[10] P. Wittek, X. Rubio-Campillo, Military reconstructive simulation in the cloud to aid battlefield excavations, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, 2012, pp. 869–874. doi:10.1109/CloudCom.2012.6427538.

[11] J. Marecki, N. Schurr, M. Tambe, Agent-Based Simulations for Disaster Rescue Using the DEFACTO Coordination System, John Wiley & Sons, Inc., 2005, pp. 281–297. doi:10.1002/047178656X.ch14.
URL http://dx.doi.org/10.1002/047178656X.ch14

[12] D. Berry, A. Usmani, J. Torero, A. Tate, S. McLaughlin, S. Potter, A. Trew, R. Baxter, M. Bull, M. Atkinson, FireGrid: Integrated emergency response and fire safety engineering for the future built environment, 2006.

[13] P. M. Mell, T. Grance, Sp 800-145. the nist definition of cloud computing, Tech. rep., Gaithersburg, MD, United States (2011).

[14] J. Ferber, Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence, Springer International Publishing, 1999.

[15] N. R. Jennings, K. Sycara, M. Wooldridge, A roadmap of agent research and development, Autonomous Agents and Multi-Agent Systems 1 (1) (1998) 7–38. doi:10.1023/A:1010090405266.
URL http://dx.doi.org/10.1023/A:1010090405266

[16] M. Wooldridge, N. R. Jennings, D. Kinny, The gaia methodology for agent-oriented analysis and design, Autonomous Agents and Multi-Agent Systems 3 (3) (2000) 285–312. doi:10.1023/A:1010071910869.
URL http://dx.doi.org/10.1023/A:1010071910869

[17] Y. Demazeau, From interactions to collective behaviour in agent-based systems, in: In: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo, 1995, pp. 117–132.

[18] Y. Wang, M. Lees, W. Cai, Grid-based partitioning for large-scale distributed agent-based crowd simulation, in: Proceedings of the 2012 Winter Simulation Conference (WSC), 2012, pp. 1–12. doi:10.1109/WSC.2012.6465161.

[19] C. Labba, N. B. ben Saoud, J. Dugdale, Towards a conceptual framework to support adaptative agent-based systems partitioning, in: Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on, 2015, pp. 1–5. doi:10.1109/SNPD.2015.7176283.

[20] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, M. Tawarmalani, Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud, SIGCOMM Comput. Commun. Rev. 40 (4) (2010) 243–254. doi:10.1145/1851275.1851212.
URL http://doi.acm.org/10.1145/1851275.1851212

[21] A. Juan-Verdejo, H. Baars, Decision support for partially moving applications to the cloud: The example of business intelligence, in: Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services, ACM, 2013, pp. 35–42. doi:10.1145/2462307.2462316.

[22] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall, 2009.

[23] T. N. Bui, C. Jones, Finding good approximate vertex and edge partitions is np-hard, Information Processing Letters 42 (3) (1992) 153 – 159. doi:http://dx.doi.org/10.1016/0020-0190(92)90140-Q.
URL http://www.sciencedirect.com/science/article/pii/0020019092901 40Q

[24] C. M. Fiduccia, R. M. Mattheyses, A linear-time heuristic for improving network partitions, in: Proceedings of the 19th Design Automation Conference, DAC '82, IEEE Press, Piscataway, NJ, USA, 1982, pp. 175–181.
URL http://dl.acm.org/citation.cfm?id=800263.809204

[25] L. F. Bittencourt, E. R. M. Madeira, Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds, Journal of Internet Services and Applications 2 (3) (2011) 207–227. doi:10.1007/s13174-011-0032-0.
URL https://doi.org/10.1007/s13174-011-0032-0

[26] W.-J. Wang, Y.-S. Chang, W.-T. Lo, Y.-K. Lee, Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments, The Journal of Supercomputing 66 (2) (2013) 783–811. doi:10.1007/s11227-013-0890-2.
URL https://doi.org/10.1007/s11227-013-0890-2

[27] F. Fittkau, S. Frey, W. Hasselbring, Cdosim: Simulating cloud deployment options for software migration support, in: 2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012, pp. 37–46. doi:10.1109/MESOCA.2012.6392599.

[28] F. Fittkau, S. Frey, W. Hasselbring, Cloud User-Centric Enhancements of the Simulator CloudSim to Improve Cloud Deployment Option Analysis, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 200–207. doi:10.1007/978-3-642-33427-6_15.
URL https://doi.org/10.1007/978-3-642-33427-6_15

[29] A. Evangelinou, M. Ciavotta, D. Ardagna, A. Kopaneli, G. Kousiouris, T. Varvarigou, Enterprise applications cloud rightsizing through a joint benchmarking and optimization approach, Future Generation Computer Systems (2016) –doi:https://doi.org/10.1016/j.future.2016.11.002.
URL http://www.sciencedirect.com/science/article/pii/S0167739X1630512X

[30] R. V. den Bossche, K. Vanmechelen, J. Broeckhove, Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds, Future Generation Computer Systems 29 (4) (2013) 973 – 985, special Section: Utility and Cloud Computing. doi:https://doi.org/10.1016/j.future.2012.12.012.
URL http://www.sciencedirect.com/science/article/pii/S0167739X12002324

[31] A. Ashraf, B. Byholm, I. Porres, Prediction-based vm provisioning and admission control for multi-tier web applications, Journal of Cloud Computing 5 (1) (2016) 15. doi:10.1186/s13677-016-0065-9.
URL http://dx.doi.org/10.1186/s13677-016-0065-9

[32] I. F. Senturk, P. Balakrishnan, A. Abu-Doleh, K. Kaya, Q. Malluhi, mit V. atalyrek, A resource provisioning framework for bioinformatics applications in multi-cloud environments, Future Generation Computer Systems (2016) –doi:https://doi.org/10.1016/j.future.2016.06.008.
URL http://www.sciencedirect.com/science/article/pii/S0167739X16301911

[33] J. Xu, M. J. Chung, Predicting the performance of synchronous discrete event simulation, IEEE Transactions on Parallel and Distributed Systems 15 (12) (2004) 1130–1137. doi:10.1109/TPDS.2004.85.

[34] C.-C. Lim, Y.-H. Low, B.-P. Gan, S. Jain, W. Cai, W. J. Hsu, S. Y. Huang, Performance prediction tools for parallel discrete-event simulation, in: Proceedings of the Thirteenth Workshop on Parallel and Distributed Simulation, PADS '99, IEEE Computer Society, Washington, DC, USA, 1999, pp. 148–155.
URL http://dl.acm.org/citation.cfm?id=301429.301469

[35] R. Solar, R. Suppi, E. Luque, High performance distributed cluster-based individual-oriented fish school simulation, Procedia Computer Science 4 (2011) 76 – 85. doi:http://dx.doi.org/10.1016/j.procs.2011.04.009.

[36] R. Solar, R. Suppi, E. Luque, Proximity load balancing for distributed cluster-based individual-oriented fish school simulations, Procedia Computer Science 9 (2012) 328 – 337. doi:http://dx.doi.org/10.1016/j.procs.2012.04.035.
URL http://www.sciencedirect.com/science/article/pii/S1877050912001561

[37] Y. Wang, M. Lees, W. Cai, Grid-based partitioning for large-scale distributed agent-based crowd simulation, in: Proceedings of the 2012 Winter Simulation Conference (WSC), 2012, pp. 1–12. doi:10.1109/WSC.2012.6465161.

[38] G. Vigueras, M. Lozano, J. Ordua, F. Grimaldo, A comparative study of partitioning methods for crowd simulations, Applied Soft Computing 10 (1) (2010) 225 – 235. doi:http://dx.doi.org/10.1016/j.asoc.2009.07.004.

[39] Y. Xu, W. Cai, H. Aydt, M. Lees, Efficient graph-based dynamic load-balancing for parallel large-scale agent-based traffic simulation, in: Proceedings of the Winter Simulation Conference 2014, 2014, pp. 3483–3494. doi:10.1109/WSC.2014.7020180.

[40] C. Márquez, E. César, J. Sorribes, Graph-Based Automatic Dynamic Load Balancing for HPC Agent-Based Simulations, Springer International Publishing, Cham, 2015, pp. 405–416. doi:10.1007/978-3-319-27308-2_33.
URL http://dx.doi.org/10.1007/978-3-319-27308-2_33

[41] J. Xu, M. J. Chung, Predicting the performance of synchronous discrete event simulation, IEEE Trans. Parallel Distrib. Syst. 15 (12) (2004) 1130–1137. doi:10.1109/TPDS.2004.85.
URL http://dx.doi.org/10.1109/TPDS.2004.85

Table 4: Comparison of existing prediction-based approaches

	Focus	Application type	Granularity	Metrics	Cloud Environment
[20]	code entities partitioning	web application	code function	- execution time - communication time -deployment costs	hybrid cloud
[30]	scheduling optimization	bag-of-tasks Applications	task	-computation costs -data transfer costs -bandwidth	hybrid cloud
[29]	cloud right-sizing and optimization	cloud applications	————	-consumed Time -compute Costs -service Efficiency	multi-clouds
[32]	optimizing multi-cloud deployment	bioinformatics applications	job	-execution Time - costs	multi-clouds
[31]	prediction and VM provisioning	multi-tier web application	component	-http requests -number of web applications -number of servers -CPU load -RAM load -response time -number of deffered, accepted and refused sessions.	amazon EC2